REMARKS/ARGUMENTS

35 USC 112

The Office objected to claim 1 as improperly using a singular verb for a plural object, and to claim 7 as being in improper independent format. Those objections are obviated by cancellation of claims 1 and 7 herein.

35 USC 102(b)

The Office rejected claims 1 and 3, 5 and 7 as being anticipated by Hunter et al. (US 5606690). Those rejections are all obviated by cancellation of the rejected claims.

35 USC 103

The Office rejected claims 2 and 6 as being obvious over Hunter in view of Major et al. (US 5455932) or Allison et al. (US 6549960), respectively. Those rejections are also obviated by cancellation.

New Claims

The new claims are all allowable over the cited art. New claims 9-19 are all dependent on claim 8, which requires that "if, after s sequential symbols in any one A of said m patterns, a sequence of no more than t sequential symbols corresponds to the first t symbols of a pattern B from amongst said m patterns, where t is at least one but less than the number of symbols in pattern B, then the (s + t) state of said unique sequential set associated with pattern A of successive transitions from the initial state also has stored therein a transition to the (t + 1) state of said unique sequential set associated with pattern B of successive transitions from the initial state." Those limitations are not found in any of the cited art.

Considerable care has been take to ensure that no new matter has been added. See chart below for claim 8.

///

Claim Text	Point References To Published Application
8. (new) A deterministic finite state automaton (FSA), responsive to an *n*-symbol alphabet, adapted to match, in parallel, *m* patterns each comprised of sequentially ordered symbols selected from said alphabet, the FSA comprising:	Fig. 4; generally, ¶ 0020 - 0023; ¶ 0107, lines 1-2; ¶ 0119.
an initial state adapted to store therein *n* nextState transitions, each transition being responsive to a respective one of said symbols in said alphabet;	Fig. 4, State 0; ¶ 0052; ¶ 0061, function createGraph, generally, and, in particular, lines 3-5; ¶ 0107, lines 2-3.
a plurality of states, each corresponding to one symbol contained at a given position in one or more of said *m* patterns, and being adapted to store therein *n* nextState transitions, each transition being responsive to a respective one of said symbols in said alphabet;	Fig. 4, States 1-4 for first of *m* patterns, where m = 2; States 5-7 for second of *m* patterns; ¶ 0052 (for inital state transistions); ¶ 0061, function createGraph, generally, and, in particular, lines 11-12 & 14; ¶ 0107, lines 3-4.
wherein: said initial state has stored therein a plurality of transitions each to a first state responsive to a first symbol of one or more of said *m* patterns;	Fig. 4, transition "f" for first pattern and transition "e" for second pattern; ¶ 0061, function createGraph, generally, and, in particular, line 13, for the first iteration for each pattern, *i.e.*, first

	symbol "f" in first pattern and first symbol "e" in second pattern; ¶ 0107, lines 2-3.
each state which corresponds to position *i* in one or more of said *m* patterns having stored therein *j* transitions to the *j* next sequential states each responsive to a particular symbol at position *i+1* in one or more of said *m* patterns, such that each of said *m* patterns is associated with one unique sequential set of successive transitions from the initial state;	Fig. 4, transitions "r", "e" and "e" for first pattern, and "e" and "l" for second pattern; ¶ 0061, function createGraph, generally, and, in particular, line 13, for the second and all subsequent iterations for each pattern, *i.e.*, symbols "r", "e" and "e" for first pattern, and "e" and "l" for second pattern; ¶ 0107, lines 3-5.
if, after *s* sequential symbols in any one A of said *m* patterns, a sequence of no more than *t* sequential symbols corresponds to the first *t* symbols of a pattern B from amongst said *m* patterns, where *t* is at least one but less than the number of symbols in pattern B, then the $(s + t)$ state of said unique sequential set associated with pattern A of successive transitions from the initial state also has stored therein a transition to the $(t + 1)$ state of said unique sequential set associated with pattern B of successive transitions from the initial state.	Fig. 4, transition "e" from State 4 of the first pattern to State 6 of the second pattern, transition "l" from State 4 of the first pattern to State 7 of the second pattern, and the recursive transition "e" at State 6 in the second pattern; ¶ 0064, function completeGraph, generally, and, in particular, lines 15 and 24, where the additional

	transitions are added using sub-function addEdges, line 7 (see, § 0065); ¶¶ 0111 - 0119.
9. (new) The FSA of claim 8 wherein all transitions stored in each of said states, other than the transitions detailed in claim 8, comprise transitions to said initial state, each responsive to a respective one of said symbols of said alphabet.	¶ 0074, function complete Dfa, to complete a dFSA; ¶ 0077, function makeDeterministic to make a ndFSA into a dFSA; ¶ 0052, lines 3-6 for array based implementations.

10. (new) A method for creating the deterministic finite state automaton (FSA) of claim 9, the method comprising the steps of:	Fig. 4; generally, ¶ 0020 - 0023; ¶¶ 0060-0068 and ¶¶ 0073-0077; ¶ 0107, lines 1-2; ¶ 0119.
(1) creating a partial graph by:	¶ 0061, lines 2-3, function CreateGraph.
(1.1) creating an initial state, and storing therein *n* nextState transitions to said initial state, each responsive to a respective one of said symbols in said alphabet;	¶ 0061, function createGraph, line 4; see, generally: ¶ 0074, function complete Dfa, to complete a dFSA; ¶ 0077, function makeDeterministic to make a ndFSA into a dFSA; ¶ 0052, lines 3-6 for array based implementations.
(1.2) selecting said first pattern;	¶ 0061, lines 2-3, function CreateGraph.
(1.3) for said selected pattern:	¶ 0061, function createGraph.
(1.3.1) selecting as a current state said initial state;	¶ 0061, function createGraph, line 6/7.
(1.3.2) selecting a first symbol in said selected pattern;	¶ 0061, function createGraph, line 8 (inherent initial functionality of "for" loop).

(1.3.3) selecting as a next state the nextState transition stored in said selected current state which is responsive to the selected symbol;	¶ 0061, function createGraph, line 9.
(1.3.4) if said next state is said initial state:	¶ 0061, function createGraph line 10/11.
(1.3.4.1) creating a next state, and storing therein *n* nextState transitions to said initial state, each responsive to a respective one of said symbols in said alphabet; and	¶ 0061, function createGraph, line 12.
(1.3.4.2) storing in said selected current state a transition to said next state responsive to said selected symbol;	¶ 0061, function createGraph, line 13; ¶ 0057, function addTransition.
(1.3.5) selecting as said current state said selected next state;	¶ 0061, function createGraph, line 15.
(1.3.6) if said selected pattern comprises at least one additional symbol, selecting a next successive symbol in said selected pattern and returning to step (1.3.3); and	¶ 0061, function createGraph, line 16 (inherent loop functionality of "for" loop initiated at line 8).
(1.3.7) if said selected pattern comprises said first pattern, selecting said second pattern and returning to step (1.3.1); and	¶ 0061, lines 2-3, function CreateGraph.
(2) completing said partial graph by:	¶ 0064, lines 1-2, function CompleteGraph.

(2.1) selecting said first pattern;	¶ 0064, lines 1-2, function Complete Graph.
(2.2) for said selected pattern:	¶ 0064, function completeGraph.
(2.2.1) selecting a first symbol in said selected pattern;	¶ 0064, function completeGraph, line 4 (symbol p_1).
(2.2.2) selecting as said current state the nextState transition stored in said initial state which is responsive to the selected symbol;	¶ 0064, function completeGraph, line 4.
(2.2.3) if said selected pattern comprises at least one additional symbol:	¶ 0064, function completeGraph, line 6 (inherent initial condition test of "for" loop).
(2.2.3.1) selecting a next successive symbol in said selected pattern;	¶ 0064, function completeGraph, line 6 (inherent functionality of "for" loop).
(2.2.3.2) selecting as said current state the nextState transition stored in said selected current state which is responsive to the selected symbol;	¶ 0064, function completeGraph, line 7.
(2.2.3.3) if a first-in-first-out queue has q states enqueued therein, where q is at least one:	¶ 0064, function completeGraph, line 9 (inherent initial condition test of "for" loop).

(2.2.3.3.1) removing from said queue the first state enqueued therein;	¶ 0064, function completeGraph, line 10 (".removeLast").
(2.2.3.3.2) selecting as a temporary state said state removed from said queue;	¶ 0064, function completeGraph, line 10.
(2.2.3.3.3) selecting as said temporary state the nextState transition stored in the temporary state selected in step (2.2.3.3.2) which is responsive to the selected symbol;	¶ 0064, function completeGraph, line 11.
(2.2.3.3.4) if said selected temporary state is not said initial state:	¶ 0064, function completeGraph, line 12/13.
(2.2.3.3.4.1) enqueueing on said queue the selected temporary state;	¶ 0064, function completeGraph, line 14 (".insertFirst").
(2.2.3.3.4.2) for each nextState transition in the selected current state which corresponds to the initial state, storing in said selected current state the corresponding nextState transition stored in said selected temporary state; and	¶ 0064, function completeGraph, line 15 (".addEdges"); ¶ 0065, function addEdges.
(2.2.3.3.4.3) if said queue has at least one of said *q* states enqueued therein, returning to step (2.2.3.3.1);	¶ 0064, function completeGraph, line 20 (inherent loop functionality of "for" loop initiated at line 9).

(2.2.3.4) selecting as said temporary state the nextState transition stored in said initial state which is responsive to the selected symbol;	¶ 0064, function completeGraph, line 21.
(2.2.3.5) if said selected temporary state is not said initial state:	¶ 0064, function completeGraph, line 22/23.
(2.2.3.5.2) for each nextState transition in the selected current state which corresponds to the initial state, storing in said selected current state the corresponding nextState transition stored in said selected temporary state; and	¶ 0064, function completeGraph, line 24.
(2.2.3.5.1) enqueueing on said queue said selected temporary state;	¶ 0064, function completeGraph, line 25.
(2.2.3.6) if said selected pattern comprises at least one additional symbol, selecting a next successive symbol in said selected pattern and returning to step (2.2.3.1); and	¶ 0064, function completeGraph, line 27 (inherent loop functionality of "for" loop initiated at line 6).
(2.3) if said selected pattern comprises said first pattern, selecting said second pattern and returning to step (2.2).	¶ 0064, lines 1-2, function CompleteGraph.
11. (new) The method of claim 10 wherein each of said states comprises an array of *n* elements, each element adapted to store a transition responsive to a respective one of said *n* symbols.	See, above; ¶ 0052.
12. The method of claim 10:	See, above.

wherein each of said states is adapted to store an action to be performed upon transition to said state; and	¶ 0049.
wherein said step (1.3.5) is further characterized as:	
(1.3.5a) selecting as said current state said selected next state; and	¶ 0061, function createGraph, line 15.
(1.3.5b) storing into said selected current state a predetermined action associated with said selected pattern.	¶ 0061, function createGraph, line 17.
13. (new) The method of claim 12 wherein each of said states comprises:	See, above.
an array of *n* elements, each element adapted to store a transition responsive to a respective one of said *n* symbols; and	¶ 0052.
an action associated with said state.	¶ 0049.
14. (new) The method of claim 13 wherein each state having an action associated therewith is adapted to perform said action in response to transition thereto during a match operation.	See, above; ¶ 0049; ¶ 0072, function nfaSearch, line 8; ¶ 0079, function dfaSearch, line 6.
15. (new) The method of claim 12:	See, above.
wherein said step (2.2.3.5.2) is further characterized as:	

(2.2.3.5.2a) for each nextState transition in the selected current state which corresponds to the initial state, storing in said selected current state the corresponding nextState transition stored in said selected temporary state; and	¶ 0064, function completeGraph, line 24.
(2.2.3.5.2b) storing into said selected current state said predetermined action associated with said selected pattern.	¶ 0064, function completeGraph, line 17.
16. (new) The method of claim 15 wherein each of said states comprises:	See, above.
an array of *n* elements, each element adapted to store a transition responsive to a respective one of said *n* symbols; and	¶ 0052.
an action associated with said state.	¶ 0049.
17. (new) The method of claim 16 wherein each state having an action associated therewith is adapted to perform said action in response to transition thereto during a match operation.	See, above; ¶ 0049; ¶ 0072, function nfaSearch, line 8; ¶ 0079, function dfaSearch, line 6.
18. (new) The FSA of claim 8 wherein:	See, above.
each final state reached by each unique sequential set of successive transitions from the initial state associated with each of the *m* patterns has stored therein an action to be performed upon transition to said final state.	¶ 0049; ¶ 0072, function nfaSearch, line 8; ¶ 0079, function dfaSearch, line 6.
19. A method of using the FSA of claim 18, the method comprising the steps of:	See, above; ¶ 0079, function dfaSearch.

receiving an input stream of symbols, at least some of which are selected from said alphabet;	¶ 0079, function dfaSearch, line 1.
presenting to said FSA each of said received symbols; and	¶ 0079, function dfaSearch, line 3.
when a sequence of symbols in said input stream is found by said FSA to match one of the said *m* patterns, performing said action.	¶ 0079, function dfaSearch, line 6.

Request For Allowance

Claims 8-19 are pending in this application. Claims 1-7 are canceled. The applicant requests allowance of all pending claims.

Request For Telephonic Interview

As should be apparent, the applicant has made every effort to provide claims that the Office could readily allow. In the event that the Examiner is inclined to require the filing of a Request for Continued Examination to pursue these new claims, the Applicant respectfully requests an opportunity to discuss the situation by telephonic interview.

Respectfully submitted,
RUTAN & TUCKER

By _____
Robert D. Fish
Reg. No. 33880

Rutan & Tucker
611 Anton Blvd., 14th Floor
Costa Mesa, CA 92626-1931
Telephone (714) 641-5100
Fax (714) 546-9035